The EVIL Machine

Encode, Visualize and Interpret the Leakage

Valence Cristiani
CEA-Leti, Univ. Grenoble Alpes
valencecristiani@gmail.com

Maxime Lecomte
CEA-Leti, Univ. Grenoble Alpes
maxime.lecomte@cea.fr

Philippe Maurine
LIRMM, Montpellier
pmaurine@lirmm.fr

ABSTRACT

Unsupervised side-channel attacks allow extracting secret keys manipulated by cryptographic primitives through leakages of their physical implementations. As opposed to supervised attacks, they do not require a preliminary profiling of the target, constituting a broader threat since they imply weaker assumptions on the adversary model. Their downside is their requirement for some *a priori* knowledge on the leakage model of the device. On one hand, stochastic attacks such as the Linear Regression Analysis (LRA) allow for a flexible *a priori*, but are mostly limited to a univariate treatment of the traces. On the other hand, model-based attacks require an explicit formulation of the leakage model but have recently been extended to multidimensional versions allowing to benefit from the potential of Deep Learning (DL) techniques. The EVIL Machine Attack (EMA), introduced in this paper, aims at taking the best of both worlds. Inspired by generative adversarial networks, its architecture is able to recover a representation of the leakage model, which is then turned into a key distinguisher allowing flexible *a priori*. In addition, state-of-the-art DL techniques require 256 network trainings to conduct the attack. EMA requires only one, scaling down the time complexity of such attacks by a considerable factor. Simulations and real experiments show that EMA is applicable in cases where the adversary has very low knowledge on the leakage model, while significantly reducing the required number of traces compared to a classical LRA. Eventually, a generalization of EMA, able to deal with masked implementation is introduced.

1 INTRODUCTION

1.1 Context

Side-Channel Analysis (SCA) is defined as the process of gaining information on a device holding a secret through its physical leakage such as power consumption [17] or Electromagnetic (EM) emanations [20]. This leakage allows an adversary to extract sensitive information such as cryptographic keys by carefully exploiting the dependencies between the secret and the side-channel data. Strategies are mainly divided into two categories: supervised and unsupervised SCA and their utilization depends on the considered threat model. In the first one, the adversary is supposed to be able to conduct a profiling step of the target, most likely on a clone device, in which she learns the leakage model of the intermediate variables and then adopts a maximum likelihood approach to recover the secret key. This includes strategies such as Gaussian template attacks [5] or deep learning profiled attacks [18]. If the model is perfectly learned during the characterization phase, these attacks are known to be optimal from an information theory point of view.

However, being able to conduct a sound profiling step is not always possible and refers to a strong threat model for the adversary. Indeed, the latter one may not be able to obtain a clone with full control on the device and even in cases where she could, template portability issues [12] may still stand in the way. In this case, the adversary can "replace" the profiling of the target by its *a priori* on the leakage model to mount what is called unsupervised SCA. Such an *a priori* usually comes from physical assumptions and is central to unsupervised SCA. Indeed, as shown in [23], there does not exist a generic unsupervised strategy that would work without requiring such an *a priori*. However, *a priori* is a very vague term that only captures the fact that the adversary has to have some knowledge about the leakage model for the attack to work. This knowledge can take many different forms.

In a first type of unsupervised SCA, known under the stochastic attacks, the *a priori* takes the form of a parametric model. For example, the well-known Linear Regression Analysis (LRA) [11] falls into this category. The advantage of such attacks is their ability to work with weak *a priori* such as the only assumption that the bits of the sensitive variable leak independently. This makes them robust and applicable in a wide range of cases.

Another type of strategy, denoted the model-based attacks in this paper, requires the *a priori* on the model to be expressed as an explicit function (a.k.a. the partition function in [9]). A famous example would be the classical Mutual Information Analysis (MIA) [14]. Some attacks of this category have recently been extended in order to use deep learning techniques [9, 22, 24] allowing to take advantage of the multidimensional treatment of the traces. It reduces at the same time the need for preprocessing with for example filtering or realignment techniques. It has been shown in [9] that such attacks can exploit a larger part of the information contained in the traces and may outperform state-of-the-art stochastic attacks. However, they suffer from two major drawbacks:

- **The choice of the partition function** which is closely related to the leakage model *a priori*. Indeed, each bit of the intermediate variable can have very different leakage behavior (especially when it comes to Electro-Magnetic (EM) measurements) and even sign inversions of their coefficients as shown in [6]. In these cases, a classical Hamming weight *a priori* may lead to unsuccessful attacks where an LRA would work.

SAC'23, March 27 –April 2, 2023, Tallinn, Estonia

ACM ISBN 978-1-4503-9517-5/23/03…$15.00
https://doi.org/xx.xxx/xxx_x

- **The time complexity of the attack** which requires as many network trainings as there are key hypotheses (which means 256 trainings when attacking a key byte). If the adversary wants to test multiple *a priori*, this issue is then amplified which leads to complex or even unpractical attacks for highly noisy target requiring a lot of traces and therefore long trainings.

1.2 Contributions

This paper presents a new unsupervised strategy, denoted the EVIL Machine Attack (EMA), which allows to use a flexible *a priori* (as in the stochastic attack) while still being able to exploit the power of deep learning. We show that it only requires one network training whatever the number of key hypotheses. Therefore, EMA overcomes the two main issues of unsupervised deep learning attacks.

The architecture of the EVIL machine is presented in section 2. It is a Generative Adversarial Network (GAN) like structure whose goal is to derive the leakage model of the target under a key assumption. The actual attack and how the number of network training is reduced to only one is described in section 3. Results on synthetic and real traces are also presented in this section. Eventually, section 4 gives an introduction to a higher-order version of the attack, when dealing with masked implementation.

2 LEARNING A LEAKAGE MODEL REPRESENTATION

One of the main difficulties of the model-based SCA is to have a sound representation of the leakage model to use it as a partition function [9]. The original idea of this work is to transfer the task of finding such a representation to a neural network without having to use any *a priori*. This representation is conditioned by a key assumption otherwise it would provide an *a priori* free strategy contradicting [23]. So to ease the reading of the paper, the correct key is first assumed to be known in this section. However, section 3 shows how such a tool can be turned into an actual unsupervised attack.

2.1 Notations and SCA framework

Random variables are represented as upper-case letters such as X. They take their values in the corresponding set $\mathcal{X}$ depicted with a calligraphic letter. Lower-case letters as x stand for elements of $\mathcal{X}$.

In order to gain information on the secret key the adversary targets the manipulation of a sensitive variable $Z \in \mathcal{Z} = \mathbb{F}_2^n$, for a given $n \in \mathbb{N}$. This variable is supposed to functionally depends on a public variable $X \in \mathcal{X} = \mathbb{F}_2^m$, for a given $m \in \mathbb{N}$, and a secret key $k^* \in \mathcal{K} = \mathbb{F}_2^m$ through the relation: $Z = g(X, k^*)$ where $g : \mathcal{X} \times \mathcal{K} \to \mathcal{Z}$ is a known function depending on the underlying cryptographic algorithm. If an hypothesis k is made on the secret key, one can define $Z_k = g(X, k)$ such that $Z = Z_{k^*}$.

For a fixed k we denote by g_k the $g(\cdot, k)$ function. All the g_k are supposed to be bijective in this paper of reciprocal g_k^{-1} (for example, in the classical first round AES case one would have $g_k(X) = \texttt{SBOX}[X \oplus k]$ of reciprocal $g_k^{-1}(Z) = \texttt{SBOX}^{-1}[Z \oplus k]$. Eventually, traces L leaking information about Z through a leakage model φ can be expressed as $L = \varphi(Z) + \mathcal{E}$ where $\mathcal{E}$ represent an independent noise variable.

2.2 Building the Network's Architecture

Classical Mutual Information (MI) attacks rank the key hypotheses with the following distinguisher:

$$\mathcal{D}(k) = \mathcal{I}\big(f(Z_k), L\big) \tag{1}$$

where $\mathcal{I}(X, Y)$ stands for the MI between X and Y and f is the partition function transforming the guessed intermediate variable. The starting point of the reasoning behind the EVIL machine is the main theorem of [9] which states that the leakage model φ belong to the set of the optimal partition functions for the following distinguishability criteria:

Theorem. *(Cristiani et al.)*

$$\varphi \in \mathcal{F}_{opt} = \underset{f:\, \mathcal{Z} \to \mathbb{R}^p}{\arg\max} \left\{ \mathcal{I}\big(f(Z_{k^*}), L\big) - \max_{k \neq k^*} \big[\mathcal{I}\big(f(Z_k), L\big) \big] \right\} \tag{2}$$

Note that traces can be multidimensional. Therefore the output domain of φ is $\mathbb{R}^p$ where p can be any positive integer.

Our goal is to derive from this theorem a neural network architecture able to produce an encoding function $\mathbf{E} : \mathcal{Z} \to \mathbb{R}$ of the leakage model φ. The main challenge is to define the network's objective function. Optimally, $\mathbf{E}$ should carry the same information as φ and therefore be a bijection of φ encoding it into one dimension[1]. Combined with Equation 2, this property could be derived as an objective for E. Indeed, since bijective transformations do not affect the MI:

$$\mathbf{E} \in \mathcal{B}(\varphi) \implies \mathbf{E} \in \mathcal{F}_{opt} \tag{3}$$

where $\mathcal{B}(\varphi)$ stands for the set of all bijections of φ. The reciprocal of Equation 3 is not formally proven even though we conjecture so. Thus, we make the hypothesis here that a function belonging to $\mathcal{F}_{opt}$ would give a valuable representation of the leakage model and try to find such a function thanks to deep learning tools. A naive idea would be to use the difference of the MI term in Equation 2 as an objective function for E. However, such an expression can hardly be used in a deep learning framework.

The first problem is the presence of the max function which is not differentiable. This problem can be solved using remark 1 of [9] which states that the theorem is still valid if one fixes $\bar{k} \in \mathcal{K} \setminus k^*$. So $\forall \bar{k} \in \mathcal{K} \setminus k^*$:

$$\varphi \in \mathcal{F}_{opt} = \underset{f:\, \mathcal{Z} \to \mathbb{R}^p}{\arg\max} \left\{ \mathcal{I}\big(f(Z_{k^*}), L\big) - \mathcal{I}\big(f(Z_{\bar{k}}), L\big) \right\} \tag{4}$$

The second problem is the computation of the MI terms which are known to be hard, especially for high-dimensional traces. However, [8] recently showed that these MI terms could be estimated using deep learning tool and more precisely a Mutual Information Neural Estimator (MINE) [1]. Therefore such a tools could be incorporated into the EVIL machine architecture in order to compute the objective function of the encoder $\mathbf{E}$. In this paper it is enough to see MINE as a black-box deep learning method that defines a network, taking as input the traces and label variable, with an objective function converging toward the MI between these two

[1] One could design an encoding function encoding the leakage model in more than one dimension. However, preliminary analyses did not show any real value of increasing the output dimension. In addition, since one-dimensional data are easier to interpret and better suited for the attack phase presented in section 3, we only focus on functions with a one-dimensional output in this paper.

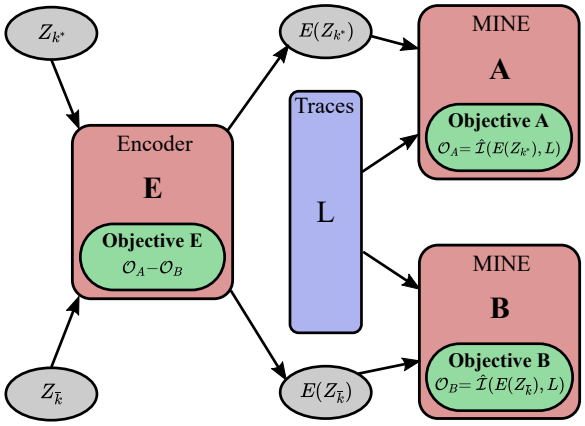


Figure 1: The EVIL Machine Architecture

variables. Further details on the theory and implementation in an SCA context can be found in [8].

Architecture. The architecture of the EVIL Machine is depicted in Figure 1. It is composed of three neural networks interacting with each other:

- An encoder **E** which takes as input a label (for instance, the value of Z_{k^*} or $Z_{\bar{k}}$) and outputs a value in $\mathbb{R}$ (its final layer is a single neuron).
- MINE **A** which objective function produces an estimation $\hat{\mathcal{I}}(E(Z_{k^*}), L)$ of $\mathcal{I}(\mathbf{E}(Z_{k^*}), L)$.
- MINE **B** which objective function produces an estimation $\hat{\mathcal{I}}(E(Z_{\bar{k}}), L)$ of $\mathcal{I}(\mathbf{E}(Z_{\bar{k}}), L)$, where $\bar{k}$ is fixed to any value in $\mathcal{K} \setminus k^*$.

The encoder **E** is applied on both Z_{k^*} and $Z_{\bar{k}}$ where $\bar{k}$ can be fixed to any value except k^*. Then $\mathbf{E}(Z_{k^*})$ and $\mathbf{E}(Z_{\bar{k}})$ are both concatenated with the traces (each trace is concatenated with its associated encoded label) and provided to the MINE networks which estimate both MI terms of Equation 4. Their difference is used as an objective function for the encoder. If all the objective functions are correctly optimized (with usual deep learning techniques), **E** should converge toward an element of $\mathcal{F}_{opt}$, potentially close to a bijection of φ. In theory, on entire execution of MINE A and B should be run after each epoch of **E** which would be very long. So we decided to mimic the idea found in the field of GANs [15] and run successively one epoch of **E** and one epoch of A and B (in parallel) hoping that this strategy accelerates the convergence without worsening the results.

2.3 Simulation Experiments

We have implemented the generic architecture described in the previous section using the TensorFlow library [13]. The precise architecture of each network is depicted in Appendix A. In order to validate the methodology and gain intuition about the network behavior, this section provides simulation experiments on synthetic traces generated with different leakage models. The idea is to observe the evolution of the output of **E** over the training epochs and to compare it to the known underlying leakage model to assess if that is converging toward a bijection of φ.

Hamming Weight Leakage Model. For the first experiment, we have generated the most basic side-channel traces composed of one sample leaking a noised version of the Hamming weight of $Z_{k^*} = SBOX[P \oplus k^*]$, with P and k^* respectively representing a plaintext and a key byte. The leakage traces can then be expressed as:

$$L = \mathrm{HW}(Z_{k^*}) + \mathcal{N}(0, 1) \tag{5}$$

with $\mathcal{N}(0, 1)$ representing the standard normal distribution. The evolution of the output of **E** is depicted in Figure 2. For each epoch, we plot $\mathbf{E}(z)$ for all $z \in \mathcal{Z}$. Each of these 256 values are colored according to their Hamming weight. The first observation is that the encoder is learning to cluster the different values of Z according to their Hamming weight. It thus correctly learns a bijection of the true leakage model. It should be noted that **E** clusters the Hamming weight classes in order which was not guaranteed by the mathematical analysis. The latter result being robust over multiple simulations, we conjectured that such a representation is "simpler" in a way and naturally emerges from the gradient descent algorithm.

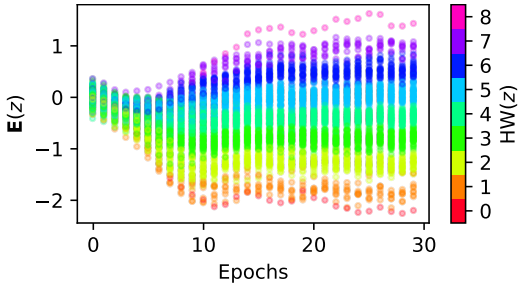


Figure 2: Evolution of the encoder's output: $\mathbf{E}(z), \forall z \in \mathcal{Z}$ versus epochs (Hamming weight leakage model).

Linear Leakage Model. To assess the EVIL machine capabilities on a more generic leakage model, we repeated the previous experiment emulating a linear leakage with respect to the bits of the sensitive variable. In this case, the leakage traces can then be expressed as:

$$L = \varphi_0(Z_{k^*}) + \mathcal{N}(0, 1) \tag{6}$$

with $\varphi_0 = \sum_1^8 \alpha_i b_i$ where b_i represents the projection on the i^{th} bit and α_i a random coefficient uniformly drawn from $[-1, 1]$. Results are depicted in Figure 3. Each point is colored according to the value of $\varphi_0(z)$. The encoder is again converging towards a meaningful representation of the leakage model. Indeed, the figure looks like a uniform rainbow which highlights the fact that the relation between $\mathbf{E}(Z_{k^*})$ and $\varphi_0(Z_{k^*})$ is quasi-linear.

Multidimensional Leakage Model. One of the main advantages of the EVIL machine is its ability to take large parts of the trace as input and to compress into one neuron multiple leakage sources. To highlight this multidimensional capability we design a simple example where the leakage is split on two samples. Each sample leaks respectively the Hamming weight of $Z^+_{k^*}$ and $Z^-_{k^*}$ which represent the four most and least significant bits of Z_{k^*}. The leakage traces can then be expressed as:

$$L = [\mathrm{HW}(Z^+_{k^*}) + \mathcal{N}(0, 1),\ \mathrm{HW}(Z^-_{k^*}) + \mathcal{N}(0, 1)] \tag{7}$$

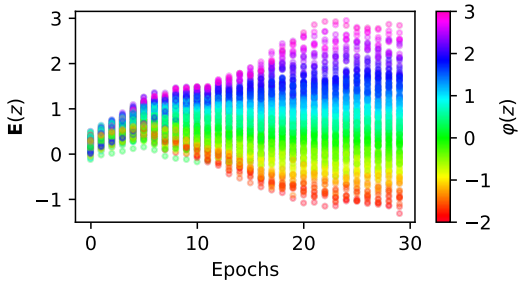
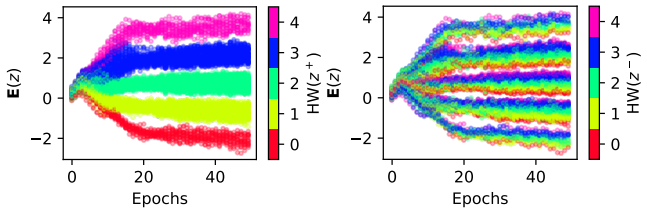


Figure 3: Evolution of the encoder's output: $\mathbf{E}(z), \forall z \in \mathcal{Z}$ versus epochs (linear leakage model).

Results are presented in Figure 4. The two plots are the same but colored differently. Figure 4a is colored according to $\mathrm{HW}(Z^+_{k^*})$ while Figure 4b is colored according to $\mathrm{HW}(Z^-_{k^*})$. It appears that the encoder uses the macro structure (the five big branches) to encode the information on $Z^+_{k^*}$ and the micro structure (the position in the branch) to encode the information on $Z^-_{k^*}$. Therefore, the encoder successfully learned a compressed version of a multidimensional leakage model. In this case it could almost be expressed as linear combination of both leakage sources: $\mathbf{E}(z) \approx \alpha \mathrm{HW}(Z^+_{k^*}) + \beta \mathrm{HW}(Z^-_{k^*})$, for some $\alpha, \beta \in \mathbb{R}$.



(a) Colored according to $HW(Z^+_{k^*})$

(b) Colored according to $HW(Z^-_{k^*})$

Figure 4: Evolution of the encoder's output: $\mathbf{E}(z), \forall z \in \mathcal{Z}$ versus epochs (multidimensional leakage model).

Non-Linear Leakage Model. Finally, to show that the EVIL machine is not limited to linear leakage models, we present an experiment with a non-linear model φ_{NL}. We used a linear leakage model φ_0 on the 7 MSBs of Z_{k^*} where the sign of the leakage is determined by the value of the LSB: $\varphi_{NL} = (-1)^{b_8} * \varphi_0$ with $\varphi_0 = \sum_1^7 \alpha_i b_i$, where α_i are random coefficient uniformly drawn from $[0, 1]$. The leakage traces can then be expressed as:

$$L = \varphi_{NL}(Z_{k^*}) + \mathcal{N}(0, 1) \tag{8}$$

Results are depicted in Figure 5. Each point is colored according to the value of $\varphi_0(z)$. In this case, it appears that the encoder learned a representation of the leakage model φ_{NL} composed of two symmetric branches, each one encoding the linear part of the model φ_0 but in the opposite direction due to the sign inversion related to the value of the LSB of Z_{k^*}. This confirms that the EVIL machine can still learn sound representations of the leakage even if the latter is non-linear.

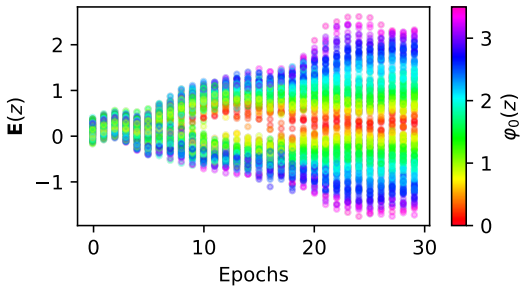


Figure 5: Evolution of the encoder's output: $\mathbf{E}(z), \forall z \in \mathcal{Z}$ versus epochs (non-linear leakage model).

This section has shown that the EVIL machine could learn a sound representation of the leakage model. Such a tool can be useful in itself, for example, for designers to gain intuition on the leakage of their devices. It does require the knowledge of the key to work, however, it is possible to use key guesses as explained in the next section which aims at turning this tool into an unsupervised attack.

3 THE EVIL MACHINE ATTACK

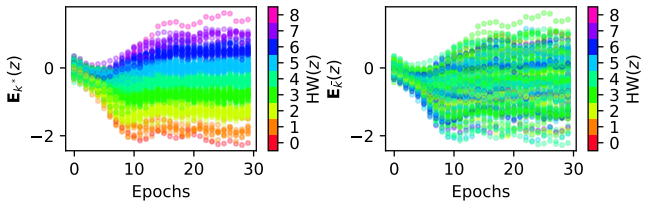
The EVIL machine presented in the previous section produces a representation of the leakage model assuming that the correct key is known. However, making hypotheses on the key allows to convert this tool into a key recovery strategy, denoted the EVIL Machine Attack (EMA). For any $k \in \mathcal{K}$ one can run an iteration of the EVIL machine producing a representation $\mathbf{E}_k$ of the leakage model φ_k representing the leakage model under the assumption that the correct key is k. All of these models φ_k mathematically exist. They are functions representing a hypothetical leakage model that would be the real one if the processed variable was the wrongly guessed $Z_k = g(X, k)$ instead of Z_{k^*}. Thus, they satisfy the following property:

$$\varphi_k(Z_k) = \varphi_{k^*}(Z_{k^*}) \tag{9}$$

They are therefore, very likely, complicated functions (with a high algebraic degree) due to cryptographic properties of the g_k functions[2] linking Z_k and Z_{k^*} together by the relation: $Z_k = g_k(g^{-1}_{k^*}(Z_{k^*}))$. The main idea of EMA is then to choose among all the possible representations of the leakage model $\mathbf{E}_k$, the simplest one, implementing a form of Occam's razor principle. This choosing step involves physical knowledge on what a leakage model should look like and this is where the adversary's *a priori* knowledge comes into play.

To illustrate the point, we repeated the Hamming weight leakage model experiment of section 2 but with a wrong key guess k_0. The evolution of $\mathbf{E}_{k_0}$ and $\mathbf{E}_{k^*}$, the one obtained with the correct key guess, are plotted in Figure 6 in order to compare them. One can visually note the difference: $\mathbf{E}_{k^*}$ is related to the Hamming weight function while $\mathbf{E}_{\bar{k}}$ seems unstructured and close to a random output. Such property can be used to define a distinguisher between key guesses as explained in subsection 3.2.

[2] Cryptographic algorithms should embed highly non-linear transformations to avoid algebraic attacks and we assume here that the targeted sensitive variable has undergone this non-linear transformation, for example, $Z = \mathtt{SBOX}(X \oplus k^*)$.



(a) Correct key **(b) Wrong key**

Figure 6: Evolution of the encoders' output for the correct and a wrong key guess.

3.1 One Training to Rule them All

Obtaining all the $\mathbf{E}_k$ can be done by running $|\mathcal{K}|$ times the EVIL machine with different key assumptions. However, this may be very long, especially for low-information scenarios requiring a lot of traces which are precisely the use cases considered in this paper. This section shows that the adversary can in fact make a first guess k_0, which may be wrong, compute $\mathbf{E}_{k_0}$, and mathematically derive $\mathbf{E}_k$ for any $k \in \mathcal{K}$ from $\mathbf{E}_{k_0}$. This method allows running only once the EVIL machine to rule all cases.

As explained in the previous section $\mathbf{E}_{k_0}$ gives a representation of φ_{k_0}, the leakage model if the correct key was k_0. For any $k \in \mathcal{K}$ Equation 9 gives:

$$\varphi_k(Z_k) = \varphi_{k^*}(Z_{k^*}) \tag{10}$$

which is also true for k_0:

$$\varphi_{k_0}(Z_{k_0}) = \varphi_{k^*}(Z_{k^*}) \tag{11}$$

So one can deduce from Equations 10 and 11 that:

$$\begin{aligned} \varphi_k(Z_k) &= \varphi_{k_0}(Z_{k_0}) \\ \varphi_k\big(g_{k_0}(g_k^{-1}(Z_k))\big) &= \varphi_{k_0}\big(g_{k_0}(g_k^{-1}(Z_{k_0}))\big) \\ \varphi_k(Z_{k_0}) &= \varphi_{k_0}\big(g_{k_0}(g_k^{-1}(Z_{k_0}))\big) \end{aligned} \tag{12}$$

which completely define φ_k from φ_{k_0}. Since $\mathbf{E}_k$ stands for a representation of φ_k, one has by analogy:

$$\mathbf{E}_k(Z_{k_0}) = \mathbf{E}_{k_0}\big(g_{k_0}(g_k^{-1}(Z_{k_0}))\big) \tag{13}$$

And since Z_{k_0} and Z live in the same set $\mathcal{Z}$, one can simplify the previous equation:

$$\mathbf{E}_k(Z) = \mathbf{E}_{k_0}\big(g_{k_0}(g_k^{-1}(Z))\big) \tag{14}$$

So one can quickly and easily deduce any $\mathbf{E}_k$ including $\mathbf{E}_{k^*}$ from the only knowledge of $\mathbf{E}_{k_0}$ with k_0 arbitrary chosen by the adversary. This reduces by a factor $|\mathcal{K}|$ (when targeting a key byte, $|\mathcal{K}| = 256$) the time complexity of the attack.

Example. *In the AES case with $g_k(P) = \texttt{SBOX}[P \oplus k]$ where P represents a plaintext byte, one can deduce $\mathbf{E}_k$ from $\mathbf{E}_{k_0}$ with the following formula:*

$$\mathbf{E}_k(Z) = \mathbf{E}_{k_0}\big(\texttt{SBOX}\big[\texttt{SBOX}^{-1}[Z \oplus k] \oplus k_0\big]\big) \tag{15}$$

3.2 About the Distinguisher

In a real attack scenario, one does not know the true leakage model so it is impossible to use the visualization technique used in the previous section (where each sample is colored according to the true leakage model) to discriminate between the key candidates. Therefore, one would need a distinguisher $\mathcal{D}$ scoring each leakage model representation $\mathbf{E}_k$ to rank the key hypotheses. The intuition behind EMA is that $\mathcal{D}$ should give a score reflecting the plausibility of the leakage representation $\mathbf{E}_k$ according to physical *a priori* on the leakage model and on which properties the latter should follow.

Assumption on the Degree of $\mathbf{E}_{k^*}$. We herein give a proposal of distinguisher for the common *a priori* stating that the leakage model φ_{k^*} should be a function from $\mathcal{Z} = \mathbb{F}_{2^n} \to \mathbb{R}^p$ such that the p coordinate functions have a low algebraic degree (bonded by a degree d). This is the multivariate version of assumption 3 (Leakage Interpolation Degree) followed in [11] for the setup of the well-known LRA. For example, a degree $d = 1$ traduces the fact that, for each time sample, all the bits of the processed variable leak independently. We propose to extend the assumption on the degree of φ_{k^*} (precisely of its coordinate functions) to the degree of its representation $\mathbf{E}_{k^*}$, produced by the EVIL machine.

Distinguisher. If this assumption is true, then one can perform a polynomial regression of order d to find the polynomial p_k of degree d minimizing the quadratic error $||\mathbf{E}_k(Z_k) - p_k(Z_k)||^2$ for all k. As explained in [11], such a regression can be seen as a linear regression by choosing an appropriate basis and is easily solvable using the least square method. A measure of fitness such as the coefficient of determination R^2 is then used as a score for each key:

$$\mathcal{D}(k) = 1 - \frac{||\mathbf{E}_k(Z_k) - p_k(Z_k)||^2}{\mathrm{var}\big(\mathbf{E}_k(Z_k)\big)} \tag{16}$$

Since the assumption on the degree of $\mathbf{E}_k$ is true only for $k = k^*$, the correct key should have the highest score leading to a successful attack.

Intuition Behind the Assumption. The justification of the assumption on the degree of $\mathbf{E}_{k^*}$ is empirical. Indeed two functions in bijection do not necessarily have the same algebraic degree. However, we observed in the experiments of section 2 that the encoder naturally converges towards a "smooth" representation of the leakage model. Our intuition is that since neural networks are composed of a succession of continuous functions it is easier for it to map two values z_1 and z_2 that have a close leakage model ($\varphi_{k^*}(z_1)$ close to $\varphi_{k^*}(z_2)$), to close encoding values $\mathbf{E}_{k^*}(z_1)$ and $\mathbf{E}_{k^*}(z_2)$. Such continuous representation has more chance of preserving the degree. In addition, in Figure 4, it seems that when dealing with multidimensional leakages, the encoder produces a linear combination of the representation of univariate leakages. Such additive behavior would also preserve the degree since by hypothesis all the coordinate functions are of degree less or equal than d.

Experiments Supporting the Assumption. One may argue that the intuitions shared in the previous paragraph come from simple experiments and may not scale with high dimensional traces containing much more leaking samples as well as uninformative

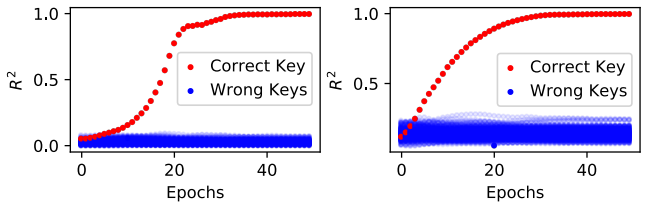
samples. Therefore we designed two experiments to assess the assumption in these harder cases. We have generated 10000 synthetic traces of dimension 1000 for both experiments. They contains 100 informative samples randomly distributed in the traces leaking information about $Z = \mathtt{Sbox}[P \oplus k^*]$ where P represents a plaintext byte uniformly drawn from $[\![0,\ 255]\!]$. In the first experiments, a linear leakage model is assigned to each sample, thus, the leakage L of each sample can be expressed as:

$$L = \alpha_0 + \sum_{i=1}^{8} \alpha_i \cdot bit_i(Z) + \mathcal{N}(0,1) \tag{17}$$

where $bit_i(Z)$ stands for the i^{th} bit of Z. The α_i are random coefficient uniformly drawn from $[-1,\ 1]$. These coefficients are redrawn for each leaking sample. The second experiment emulates quadratic leakages (with no linear coefficients) such that for each sample:

$$L = \alpha_0 + \sum_{i=1}^{8} \sum_{j=i+1}^{8} \alpha_{i,j} \cdot bit_i(Z) \cdot bit_j(Z) + \mathcal{N}(0,1) \tag{18}$$

In both experiments, the 900 non-informative samples follow a normal distribution $\mathcal{N}(0,1)$. We ran the EVIL machine assuming $k_0 \neq k^*$ and derived all the $\mathbf{E}_k$ as explained in subsection 3.1. Figure 7 reports the evolution of $\mathcal{D}(k)$ for each k computed with linear and quadratic regression respectively for the first and second experiments. In both cases, the coefficient of determination converges toward 1 for the correct key. This means that after some epochs, $\mathbf{E}_{k^*}$ can be perfectly regressed by a linear/quadratic model. These two results support the assumption on the degree of $\mathbf{E}_{k^*}$ regarding the maximum degree of the coordinate function of the leakage model φ_{k^*}. In addition, this is not at all the case of $\mathbf{E}_k$ for $k \neq k^*$ leading to a high distinguishability between the correct and the wrong keys.



(a) Linear leakages **(b) Quadratic leakages**

Figure 7: Evolution of the distinguisher $\mathcal{D}(k)$ versus epochs for all the key candidates.

3.3 Attack Description

The steps required to perform the EMA are summarized hereafter.

(1) Choose any k_0 and run the EVIL machine for n_e epochs with k_0 as a key assumption. This gives a representation $\mathbf{E}_{k_0}^{(e)}$ for each epoch e.

(2) Derive $\mathbf{E}_k^{(e)}$ for all e and k using the formula:

$$\mathbf{E}_k^{(e)}(Z_{k_0}) = \mathbf{E}_{k_0}^{(e)}\left(g_{k_0}(g_k^{-1}(Z_{k_0}))\right) \tag{19}$$

(3) Compute for all e and k the distinguishing score:

$$\mathcal{D}^{(e)}(k) = 1 - \frac{||\mathbf{E}_k^{(e)}(Z_k) - p_k(Z_k)||^2}{\mathrm{var}\left(\mathbf{E}_k^{(e)}(Z_k)\right)} \tag{20}$$

(4) Derive a single score for each key:

$$\mathcal{D}(k) = \max_{n_s \leq e \leq n_e} \mathcal{D}^{(e)}(k) \tag{21}$$

(5) Rank the keys according to their distinguishing score.

The n_s and n_e parameters are hyper-parameters set by the adversary. The purpose of n_s is to not take into account potential early fluctuation where the networks might be not stable yet. This turned out to be useful for very noisy situations (for our experiments, we used $n_s = 50$ and $n_e = 100$).

3.4 Experimental Results

EMA uses the classical LRA distinguisher on the encoder's output whose purpose is to encapsulate all the informative components of the traces. The classical LRA computes the regression directly on the traces in a univariate way (selecting the maximum score along all samples as a distinguisher). This section provides simulations and real case experiments aiming at comparing both attacks in order to assess to what extent the deep learning step is valuable. Both attacks are unsupervised and require very limited knowledge on the leakage model of the target. The main advantage of EMA over LRA is the multidimensional treatment of the traces. The latter offers the capability to potentially exploit multiple leakage sources at the same time while reducing the need of preprocessing techniques thanks to the power of DL techniques. For example, convolutional layers may handle desynchronized traces [3]. To confirm that these advantages translate into objective improvements in terms of noise resilience or number of traces required, we present hereafter two experiments.

Experiments on Synthetic traces. For the first experiment, we generated multidimensional synthetic traces subject to desynchronization. These traces contains 20 informative samples, leaking information about $Z = \mathtt{Sbox}[P \oplus k^*]$ trough linear leakage models, with randomly chosen coefficients, to which we added Gaussian noise with a standard σ. Traces contains also 80 uninformative samples and are artificially desynchronized. Each trace is shifted by a random number sh uniformly drawn from $[\![0, 50]\!]$. The precise generation procedure is depicted in Algorithm 1. We ran the classical LRA and the EMA following the procedure explained in subsection 3.3 for each value of $0 \leq \sigma \leq 25$. Results are presented in Figure 8a. Each point represents the average rank of the correct key computed over 100 independent experiments realized with a given value of σ. It appears that the EMA is way more resilient to the noise added to the traces resulting in successful attacks where the LRA fails. We highlight the fact that with such leakage models, (linear model with random coefficient from $[-1,\ 1]$) it would be hard to run an effective model-based attack since it would require an explicit model assumption.

Experiments on Real Traces. In order to validate our methodology on a real case scenario, we have acquired one million AES traces from a cortex A7-based System-on-Chip (SoC) running a

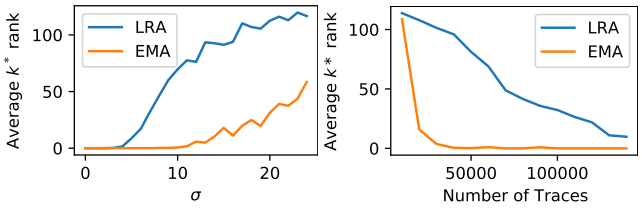
Algorithm 1 Generate Traces

Input: σ
Output: L, **a** $(10k, 100)$ **array**

$P \leftarrow$ Draw 10k plaintext uniformly from $[\![0, 255]\!]$
$Z \leftarrow \mathsf{Sbox}[P \oplus k^*]$
$L \leftarrow$ Draw a (10k, 100) array from a Gaussian $\mathcal{N}(0, \sigma^2)$
$R \leftarrow$ Draw a (20, 8) array of random coefficients from $\mathcal{U}_{[-1,1]}$

for $1 \leq i \leq 10$k **do**
  for $1 \leq j \leq 20$ **do** ▷ Add leakage one every 5 samples
    $L[i, 5 * j] \leftarrow L[i, 5 * j] + \sum_{l=1}^{8} R[j, l] \cdot bit_l(Z[i])$
  end for
end for
for $1 \leq i \leq 10$k **do**
  $sh \leftarrow$ Draw a random integer uniformly from $[\![0, 50]\!]$
  $L[i] \leftarrow \mathrm{Roll}(L[i], sh)$ ▷ Apply the jitter (Roll is a function shifting the array by sh, in a looping way)
end for
return L

Linux OS. The AES implementation comes from the OpenSSL library [21]. We chose to run the analysis on a very noisy SoC subject to desynchronization, thus, offering an interesting case. Traces[3] are composed of 1500 samples corresponding to the execution of the first round Sbox. Guessing entropies of LRA and EMA are presented in Figure 8b. Each point represents the average rank of the correct key computed over 100 experiments in which traces are chosen randomly from the dataset. It appears that the EMA converges toward the correct key faster than the LRA ranking it always at the first position with 40k traces where the LRA does not with 150k traces. This confirms that the EMA may be worth considering in very noisy scenarios, offering an unsupervised deep learning-based attack requiring only one network training and very little knowledge on the device leakage model.



(a) Synthetic traces **(b) Real traces (A7 based SoC)**

Figure 8: Guessing entropies of EMA and LRA.

4 INTRODUCTION TO HIGHER-ORDER GENERALIZATION

One of the most widely used countermeasures to prevent instantaneous leakages and mitigate first-order SCA is to mask the sensitive data using secret sharing techniques [4]. The idea is to split each sensitive intermediate value z, into d shares: $(z_i)_{1 \leq i \leq d}$. The $d-1$ shares $z_2, ..., z_d$ are randomly chosen and the last one, z_1 is processed such that:

$$z_1 = z * z_2 * \cdots * z_d \tag{22}$$

for a group operation $*$ of $\mathcal{Z}$. This led to the emergence of the so called higher-order attacks which combine multiple samples from the same traces leaking information from each share to deduce information on Z. The EVIL machine is, by nature, multidimensional and may therefore be extended to masked implementations. Indeed, it has been shown in [8] that MINE can automatically recombine samples to find information in masked cases. It would therefore be possible to run straightforwardly the EVIL machine on masked implementations. However, the main theoretical problem is that in masked case the leakage model can not be expressed as a deterministic function $L = \varphi(Z)$ with some noise. Thus, there is no equivalent of Equation 2 for masked cases. A natural question is then to ask towards what kind of representation should the encoder typically converge.

4.1 Encoder's Output and Joint Moments

When dealing with masked implementations, the leakage variable is not separable into a deterministic and a noise part anymore. Instead, traces correspond to realizations of a leakage variable L coming from a stochastic process $\mathcal{S}$, $Z \xrightarrow{\mathcal{S}} L$ which encapsulates the random choice of the masks and the leakage model of each share. L follows a multivariate probability distribution which, as any distribution, is completely determined by the list of all its joint moments. First-order SCA exploit information in the first-order moment: the leakage model being the mean per class:

$$\varphi(z) = \mathbb{E}[L \mid Z = z] \tag{23}$$

For a masked implementation the discriminating information, if it exists, is necessarily hidden in higher-order joint moments since lower-order leakages are prevented by masking. Therefore, the idea of higher-order attacks [7, 10, 19] is to exploit the lowest-order[4] centered joint moment containing information, thus replacing Equation 23 by the joint moment per class. For a d-order masked implementation and $(L_1, \ldots, L_d)$ representing one leakage sample for each share, such a joint moment can be expressed as:

$$jm_L(z) = \mathbb{E}\left[\prod_{i=1}^{d}(L_i - \mu_{L_i} \mid Z = z)\right] \tag{24}$$

where μ_{L_i} stands for the expectation of L_i. For example, the classical second-order CPA [19] combines samples with the centered product function which happens to be the covariance, *a.k.a.* the second order joint moment.

Hypothesis on the Encoder's Output. By analogy with the unmasked case where the encoder of the EVIL machine converged towards a representation of φ (Equation 23) our hypothesis is that, for masked implementations, the encoders will converge towards a representation of jm_L (Equation 24). Our intuition is that $jm_L(Z)$ is a sound way to encode Z to keep information between $jm_L(Z)$ and L while limiting the information $\mathcal{I}\left(jm_L(Z_{\bar{k}}), L\right)$ computed with a wrong key candidate $\bar{k}$.

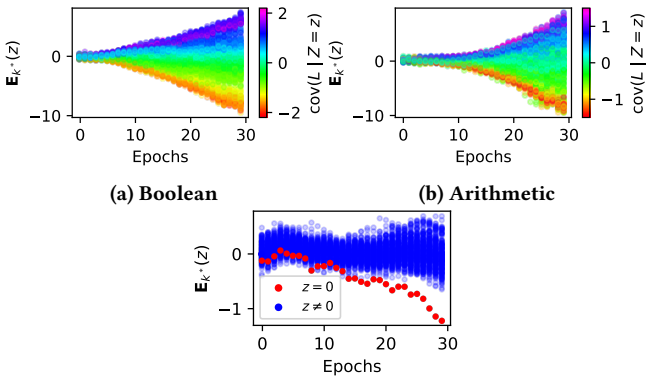
[3] For reproducibility reasons and potential use in the SCA community, the dataset with the associated labels is accessible on this link: `removed for blind reviews`.

[4] Higher-order moments being harder to estimate due to noise amplification.

Experiment Supporting the Hypothesis. To validate this hypothesis, we present experiments on masked synthetic traces for three different second-order masking schemes: Boolean, arithmetic and multiplicative. The group operations are respectively the xor $\oplus$, the addition over $\mathbb{Z}/n\mathbb{Z}$ $+[n]$ and the multiplication over the Galois field $\otimes$. The multiplication by m being reversible for $m \neq 0$ the multiplicative mask has to be chosen in $\mathbb{F}_{2^n}^*$. In addition, the multiplicative scheme is biased since $Z = 0$ implies $Z \otimes m = 0$ which induces a first-order leakage. For each masking scheme, we have generated 50k traces containing two samples representing the leakage of each share, Z_1 and Z_2, through a linear model. The leakage L can be expressed as:

$$L = \left[\sum_{i=1}^{8} \alpha_i bit_i(Z_1) + \mathcal{N}(0,1), \sum_{i=1}^{8} \beta_i bit_i(Z_2) + \mathcal{N}(0,1) \right] \quad (25)$$

where the α_i and β_i coefficients are uniformly drawn from $[-1, 1]$. We have run the EVIL machine as explained in section 2 and present the result in Figure 9. For the Boolean and arithmetic schemes, each point is colored according to the theoretical covariance per class: $\text{cov}(L \mid Z = z)$ which is computable knowing the α_i and β_i. It appears that the encoder is converging towards a smooth representation of covariance per class (the second-order joint moment), thus, supporting the hypothesis. For the multiplicative scheme, it seems that the encoder learned to distinguish the class $Z = 0$ from the others, confirming that it is focusing on lower-order leakages, if it exists, in priority.

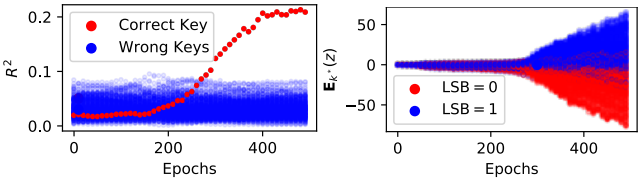


(a) Boolean (b) Arithmetic

(c) Multiplicative

Figure 9: Evolution of the encoder's output: $\mathbf{E}(z), \forall z \in \mathcal{Z}$ on masked synthetic traces (linear leakage model of the shares).

About the Distinguisher. Attacks with flexible *a priori* have been extended to masked implementation mainly from two works: [7, 10]. In [10], authors presents a higher-order version of the LRA, the HO-LRA, by proposing to perform the LRA on the estimated joint moment per class instead of the raw traces. In [7], authors highlight some limitations of the previous works and propose to directly regress the leakage model of each share with the Joint Moment Regression (JMR) strategy. Both attacks use an estimation of the jm_L function calculated under a key assumption as input to the method. Therefore we argue that these attacks could be conducted on the output of the encoders of the EVIL machine. Indeed, as suggested



(a) Evolution of the distinguisher $\mathcal{D}(k)$ versus epochs for all the key candidates. **(b) Evolution of the encoder's output: $\mathbf{E}(z), \forall z \in \mathcal{Z}$ colored according to the LSB of z.**

Figure 10: Attack results on ASCAD with 10k traces.

in the previous section, $\mathbf{E}_{k^*}(Z)$ could be seen as a representation of the jm_L function under the correct key assumption. The trick introduced in subsection 3.1 still apply and all the $\mathbf{E}_k$ can be derived from only one network training.

Again, the main advantage of the EMA is that these representations benefit from all the deep learning techniques. In addition, in classical d-variate attacks, the adversary should select the d samples from the traces to combine and perform the joint moment estimation. In a complete black-box setting, with no information on the point of interests, the latter has to enumerate all the possible d-(upplet) samples and select the best one which can be very long and scale exponentially with d. The EMA does not suffer from the same problem since the whole (or at least a big part of) traces can be fed to the network.

4.2 A Practical Case on ASCAD

As a proof of concept of the higher-order version of the EMA, this section presents an attack on the public ASCAD dataset [2]. It is a common set of side-channel traces, introduced for research purposes on deep learning-based side-channel attacks. The targeted implementation is a software AES, protected with a first-order Boolean masking, running on an 8-bit ATMega8515 board. Since JMR and the HO-LRA are equivalent for the second-order Boolean case, as shown in [7], we adopt the HO-LRA strategy which is simpler. It just consists in conducting an LRA on the estimated covariance or on $\mathbf{E}_k$ for the EMA case. It is therefore the exact same attack as for the first-order case described in subsection 3.3. We ran the EMA on 10k of the training traces which consist of 700 samples targeting the first Sbox. Results are presented in Figure 10a. The attack is successful with a clear distinguishability for the correct key.

In order to identify which features of the traces has been exploited by the EVIL machine (with the correct key assumption) we plotted the evolution of the encoder's output in Figure 10b and colored each point of according to the value of the LSB. Indeed, [9] showed that the higher-order leakage of ASCAD traces is mostly related to the value of the LSB of Z. It appears that after 200 epochs (when the attack begins to work) the encoder actually learned to split the two classes related to the LSB illustrating the interpretability of the encoder's output.

To give an order of magnitude, running the full attack (with the 500 epochs) took approximately 5 minutes on a workstation embedding a Tesla V100.

5 CONCLUSION

This paper presents a new unsupervised side-channel key recovery strategy denoted the EVIL Machine Attack (EMA). It is built on a GAN-like structure that aims at converging towards a representation of the leakage model of the device under a key assumption. Occam's razor's principle allows to derive from this tool an actual attack enabling the utilization of deep learning's potential while requiring very few knowledge on the leakage model. For example, a simple linear leakage assumption is enough as in a classical LRA. It requires only one network training where classical unsupervised deep learning strategies require as many trainings as key guesses (256 when targeting a key byte). Thus EMA reduces significantly the time complexity while being more flexible in the leakage model *a priori* than unsupervised DL based attack. This may lower the barrier for conducting such kind of attacks in practice. Simulations and real cases experiments suggest that the additional information brought by the multidimensional treatment of traces is beneficial and translates into more data-efficient attacks than the classical LRA. For example, it lower by more than 400% the number of required traces in the real case scenario targeting the openSLL implementation of the AES running on a SoC. Eventually, the last part of the paper is dedicated to an introduction of higher-order generalizations of the strategy, replacing the leakage model estimation with the joint moment estimation.

REFERENCES

[1] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. 2018. MINE: Mutual Information Neural Estimation. arXiv:cs.LG/1801.04062

[2] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. 2018. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *ANSSI, France & CEA, LETI, MINATEC Campus, France.* (2018).

[3] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. 2017. Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures. In *Cryptographic Hardware and Embedded Systems – CHES 2017*, Wieland Fischer and Naofumi Homma (Eds.). Springer International Publishing, Cham, 45–68.

[4] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. 1999. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *Advances in Cryptology — CRYPTO' 99*, Michael Wiener (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 398–412.

[5] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. 2002. Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*.

[6] Valence Cristiani, Maxime Lecomte, and Thomas Hiscock. 2019. A Bit-Level Approach to Side Channel Based Disassembling. In *CARDIS 2019*. Prague, Czech Republic. https://hal.archives-ouvertes.fr/hal-02338644

[7] Valence Cristiani, Maxime Lecomte, Thomas Hiscock, and Philippe Maurine. 2022. Fit The Joint Moments - How to Attack any Masking Schemes. Cryptology ePrint Archive, Paper 2022/927. https://eprint.iacr.org/2022/927 https://eprint.iacr.org/2022/927.

[8] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. 2020. Leakage Assessment Through Neural Estimation of the Mutual Information. In *Applied Cryptography and Network Security Workshops*. Springer International Publishing.

[9] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. 2021. Revisiting Mutual Information Analysis: Multidimensionality, Neural Estimation and Optimality Proofs. Cryptology ePrint Archive, Report 2021/1518.

[10] Guillaume Dabosville, Julien Doget, and Emmanuel Prouff. 2013. A New Second-Order Side Channel Attack Based on Linear Regression. *IEEE Trans. Comput.* 62, 8 (2013), 1629–1640. https://doi.org/10.1109/TC.2012.112

[11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. 2012. Univariate side channel attacks and leakage modeling. *Journal of Cryptographic Engineering* 1 (04 2012), 123–144. https://doi.org/10.1007/s13389-011-0010-2

[12] M. Elaabid and Sylvain Guilley. 2012. Portability of templates. *Journal of Cryptographic Engineering* 2 (05 2012). https://doi.org/10.1007/s13389-012-0030-6

[13] Martín Abadi et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[14] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. 2008. Mutual Information Analysis. In *Cryptographic Hardware and Embedded Systems – CHES 2008*, Elisabeth Oswald and Pankaj Rohatgi (Eds.). Springer Berlin Heidelberg.

[15] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. https://doi.org/10.48550/ARXIV.1406.2661

[16] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. arXiv:cs.LG/1412.6980

[17] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Annual International Cryptology Conference*.

[18] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. 2019. A Comprehensive Study of Deep Learning for Side-Channel Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020 (2019).

[19] E. Prouff, M. Rivain, and R. Bevan. 2009. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Trans. Comput.* 58, 6 (2009), 799–811. https://doi.org/10.1109/TC.2009.15

[20] Jean-Jacques Quisquater and David Samyde. 2001. Electromagnetic analysis: Measures and counter-measures for smart cards.

[21] The OpenSSL Project. 2003. OpenSSL: The Open Source toolkit for SSL/TLS. (April 2003). www.openssl.org.

[22] Benjamin Timon. 2019. Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019, 2 (Feb. 2019), 107–131. https://doi.org/10.13154/tches.v2019.i2.107-131

[23] Carolyn Whitnall, Elisabeth Oswald, and François-Xavier Standaert. 2014. The Myth of Generic DPA...and the Magic of Learning. In *Topics in Cryptology – CT-RSA 2014*, Josh Benaloh (Ed.). Springer International Publishing, Cham, 183–205.

[24] Chi Zhang, Xiangjun Lu, and Dawu Gu. 2021. Binary Classification-Based Side-Channel Analysis. In *2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. 1–6. https://doi.org/10.1109/AsianHOST53231.2021.9699563

A NETWORKS ARCHITECTURE

Figure 11 and Figure 12 show the network architectures used for all the the experiments of this paper. The optimizer used is Adam [16] with default parameters. The convolutional and fully connected layers use the exponential linear unit (ELU) as activation function.

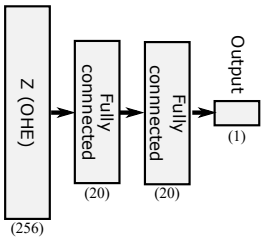


Figure 11: Architecture of the encoder E

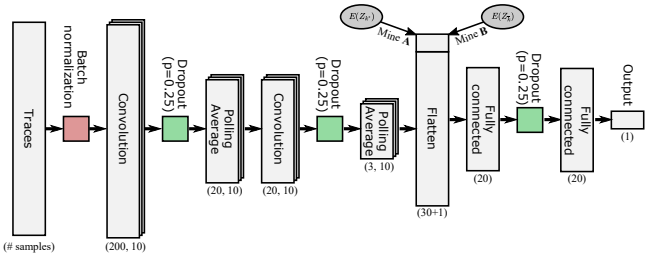


Figure 12: Architecture of MINE A and B